l'apple épluché

Nous avons reçu deux lettres de lecteurs qui se sont lancés à l'assaut du DOS 3.2 et du DOS 3.3. Voici le point où ils en sont arrivés dans leurs recherches.

Nous espérons que ces quelques informations vous serviront de point de départ et que vous serez nombreux à nous faire part de vos recherches et de vos résultats que nous ne manquerons pas de publier dans cette rubrique.

Vincent Luc

En direct du DOS 3.2

Pour le DOS 3.2 le processus de chargement du disque est fait en 3 étages. La première est effectuée par les prom P5A du contrôleur qui lit le secteur (0,0) de la disquette et le charge en page 8 ($800 à $8FF). On exécute alors le code ainsi chargé à partir de $801. Ce programme constitue la deuxième étape du chargement et utilise comme paramètre les deux derniers octets.

$8FF : $48. Cet octet indique le nombre de secteurs à lire. Pour trouver la valeur exacte, il faut diviser le contenu de $8FF par 8. Dans le cas courant où la valeur de $8FF est $48, le nombre de secteurs à charger est égal à 9. La deuxième étape chargera donc les secteurs (0,1) à (0,9). L'endroit où ces secteurs sont chargés en MEV est donné par l'octet $8FE qui contient le numéro de page moins 1. Dans le cas d'un Apple 48 K où le SED vient se placer jusqu'en $B7FF, la valeur de $8FE est $B6. Cette seconde étape permet notamment de charger la routine RWTS utilisée dans la troisième étape pour charger les autres secteurs.

En DOS 3.2, les informations sont stockées sur disquettes par paquet de 5 bits. Chaque secteur est composé comme suit :
. octets de temporisation séparant les différents secteurs (valeurs $FF). Il y en a au moins 16 entre chaque secteur.
. Hoc adresse avec
* marques de début : $D5.AA.B5
* numéro de volume, n° piste, n° de secteur
* contrôle de parité
* marques de fin : $AA-EB-FF
. octets de temporisation
. Hoc données avec
* marques de début : D5-AA-AD
* données
* contrôle de parité
* marques de fin : DE-AA-EB

A noter que seules les données sont normalement accessibles à l'utilisateur.

Jean-François Duvivier

Dans l'intimité du dos 3.3

Depuis que j'ai pu mettre la main sur un lecteur de disquettes, je suis hanté par de nombreuses questions. Je ne supporte pas l'idée que l'on puisse me dire sans explication : « Si vous mettez cette PROM, vous avez 16 secteurs par piste, avec celle-là, vous en avez 13 ».

En plus il y a deux PROMS. L'une d'entre elles contient un programme pour le 6502, mais l'autre ? J.ai cru longtemps que ces deux PROMS se partageaient les mêmes adresses. Je me trompais : la seconde est utilisée de façon interne au contrôleur et n'intéresse jamais le microprocesseur.

C'est en étudiant le programme contenu dans la PROM adressable (la P5A pour le DOS3.3) que j'ai compris comment les choses se passent. Il reste encore beaucoup d'inconnues, mais j'ai déjà quelques renseignements intéressants sur le processus d'initialisation du SED (le BOOT) et sur la façon dont le disque est lu.

Nous supposerons pour simplifier que nous avons un Apple de 48 K avec un disque en fente n° 6 contenant une disquette « esclave ». Une disquette esclave est initialisée (elle contient le SED) mais ne peut servir à mettre en route que des systèmes d'une taille mémoire déterminée.

A chaque fente sont affectées des zones mémoires particulières. Pour la fente 6, nous avons :
$COEO-$COEF contrôle du disque
$C600-$C6FF PROM du contrôleur

Les 16 octets de contrôle du disque sont connectés aux circuits du contrôleur. Je ne connais pas encore la fonction de toutes ces commandes mais voici quelques indications :
$COEO-$COE7 contrôle de la tête c'est l'ordre des positions paires et impaires qui intervient.
$COE8 arrêt du moteur
$COE9 mise en route du moteur
$COEA sélection lecteur 1
$COEB sélection lecteur 2
$COEC lecture
$COED ? (écriture ?)
$COEE mode lecture
$COEF ? (mode écriture ?)

Toutes ces commandes (sauf $COED et $COEF pour lesquelles j'en suis réduit aux conjectures) sont activées en lisant la zone correspondante. En lecture, la valeur lue en $COEC a toujours le bit le plus fort à 1 pour une lecture valide (comme pour le clavier).

Ces lectures n'ont qu'un rapport très lointain avec les données en octets qui nous intéressent. En effet le programme de lecture contenu dans la PROM du DOS3.3 suppose que les données fournies par le contrôleur ne peuvent prendre que 66 valeurs différentes ! Deux de ces valeurs ($AA et $D5) servent à la synchronisation ; restent 64 valeurs, soit 6 bits par lecture.

Les 64 valeurs que l'on peut obtenir du contrôleur ne sont pas quelconques. Il faut supprimer le bit de signe ; sur les 7 bits qui restent, il faut que deux bits à 1 se suivent au moins une fois et que deux bits à 0 se suivent au plus une fois. Plus précisément, si la valeur lue est 1b6 b5b4b3b2b1b0, on peut vérifier que (sur 6 bits) :

b6b5b4b3b2b1 ∧ b5b4b3b2b1b0
est différent de 0
b6b5b4b3b2b1 ∨ b5b4b3b2b1b0
a au plus un bit à 0

C'est d'ailleurs la méthode utilisée dans le programme de la PROM pour calculer la table lecture valeur (de 0 à $3F). Cette table est créée dans la zone $356-$3D5, mais son adresse de base est $2D6 car toutes les lectures seront supérieures à 127. Pour en terminer avec les valeurs binaires, signalons que les marques de synchronisation sont
AA (1) 0101010
D5 (1) 1010101
ce qui ne saurait être le fait du hasard.

Donc le DOS3.3 lit des paquets de 6 bits. Ce qui est stupéfiant, c'est que le DOS3.2, lui, lit des paquets de 5 bits ! Quel rôle joue la seconde PROM là-dedans ? Je n'en sais rien. Je soupçonne qu'elle contient un programme pour les circuits logiques de la carte (jusqu'où le logiciel ne va-t-il pas se nicher ?).

Par contre la PROM adressable a pu me livrer tous ses secrets. Voici ses principales fonctions (les adresses sont relatives au début de la PROM, soit Cs00 si vous êtes en fente s).
$00-$01 Création de la table des valeurs lues en $356-$3D5
$21-$38 Initialisation.
$3B-$50 Positionnement de la tête en piste 0.
$52-$5A Initialisation du tampon de lecture ($800).
$5C-$D3 Lecture brute.
$D5-$E9 Affinage lecture.
$EB-$F6 Bouclage.
$F8 Saut en $801.

La partie lecture du programme est assez compliquée. J'en ai déduit que chaque secteur est lu sous la forme suivante :
Synchronisation
$D5 $AA $96
En-tête
6 lectures pour n° volume, n° piste, n° secteur
Synchronisation
$D5 $AA $AD
Information
342 lectures
Somme de contrôle
1 lecture

Remarques : Les valeurs $96 et $AD sont celles qui figurent sur la piste 0, elles sont peut-être différentes ailleurs. Chaque lecture n'a aucun sens isolément. Il faut la transformer en valeur de 0 à 63 par la table en $2D6 puis prendre le OU exclusif

de cette valeur et de toutes les précédentes. La dernière lecture ramène cette « somme » à 0 si la lecture est correcte.

En lisant un secteur, nous obtenons 342 fois 6 bits, ce qui correspond bien à 256 octets (plus 4 bits). Pour cela nous devons faire 355 lectures. Pour une piste de 16 secteurs, cela fait 5680 lectures en tout. Si nous supposons que le DOS3.2 travaille de la même façon, avec des lectures de 5 bits, il lui faut 423 lectures par secteur, soit 5499 lectures par piste de 13 secteurs. La différence essentielle entre les deux est donc cette lecture par 5 ou 6 bits, car le nombre total de lectures par piste est sensiblement le même. Accessoirement, puisque la disquette fait 5 tours par seconde, il y a en moyenne 35 microsecondes entre deux lectures (à moins de gros vides entre les secteurs).

Maintenant, que représentent ces 342 valeurs de 6 bits ?

Au moment où elles sont lues, les 86 premières valeurs sont stockées de $355 à $300 (en descendant). Les 256 suivantes sont mises en page 8 en ordre ascendant. Après vérification de la lecture par la somme de contrôle, les valeurs définitives sont rétablies en page 8.

Les valeurs du premier groupe contiennent chacun les deux derniers bits de 3 octets. Les valeurs du second groupe contiennent les 6 derniers bits de l'octet dont elles occupent la place.

Par exemple, si les octets $00, $56, et $AC du secteur sont
$00 a7a6a5a4a3a2a1a0
$56 b7b6b5b4b3b2b1b0
$AC c7c6c5c4c3c2c1c0

nous avons après le chargement en pages 3 et 8
$355 c0c1b0b1a0a1 1e lecture
$800 a7a6a5a4a3a2 87e lecture
$856 b7b6b5b4b3b2 173e lecture
$8AC c7c6c5c4c3c2 239e lecture

Le programme de $D5 à $E9 sur la PROM remet tout ça en place !

Une fois que le premier secteur (0,0) est placé en page 8, le programme regarde l'octet $800 pour savoir si d'autres secteurs doivent être lus. Dans notre cas, cet octet contient 1 et on passe directement à l'exécution du programme lu, à partir de $801.

Sans rentrer dans les détails, ce programme utilise la partie lecture de la PROM pour charger, de $B600 à $BFFF, 10 secteurs du $ED, qui contiennent en particulier la routine « officielle » de lecture/écriture, RWTS. Pour cela il utilise les deux derniers octets du secteur.
$8FE = B6 (adresse de chargement)
$8FF = 09 (nombre de secteurs —1)

Le contrôle est ensuite transféré à $B700 pour la lecture du reste du SED.

A partir de là, c'est l'inconnu, pour l'instant en tout cas. Car une question fascinante se pose : pourquoi ne pas utiliser des fichiers en codage 6 bits ? 64 valeurs suffisent pour des fichiers alphanumériques et nous y gagnerions un tiers de capacité et un temps d'accès plus rapide. A suivre donc et si vous avez des idées...

Guilhem Bouteloup

Table de lecture du DOS 3.3

Cette table est créée par le programme de la PROM pour transformer les valeurs fournies par le contrôleur en valeurs de 6 bits.

La table est donnée ici relativement à son adresse de base, $2D6.
MS : synchronisation.
Les octets qui ne font pas partie de la table ne sont pas modifiés.

G.B.

	0 8	1 9	2 A	3 B	4 C	5 D	6 E	7 F
90							00	01
98			02	03		04	05	06
A0							07	08
A8			MS	09	0A	0B	0C	00
B0			0E	0F	10	11	12	13
B8		14	15	16	17	18	19	1A
C0								
C8				1B		1C	1D	1E
D0				1E		MS	20	21
D8		22	23	24	25	26	27	28
E0						29	2A	2B
E8		2C	2D	2E	2F	30	31	32
F0			33	34	35	36	37	38
F8		39	3A	3B	3C	3D	3E	3F

l'a.b.c. du p.e.t.

Notation musicale

Ce petit programme utilise une imprimante CBM 3022 (ou les modèles équivalents à espacement réglable) et il permet aux joueurs de flûtes à bec de noter des doigtés correspondant aux notes sur la portée musicale. Il fait appel à de nombreuses possibilités de l'imprimante.

Examinons en détail les diverses parties :

. Ligne 110 : ouverture de trois canaux pour l'imprimante :
Le canal 4 qui imprime les données telles qu'elles sont envoyées,
Le canal 1 qui imprime les données suivant un format à préciser
Le canal 6 qui détermine l'espacement entre les lignes

. Ligne 120 : Création de la chaîne qui engendre le caractère programmable

. Ligne 130 : Envoi à l'imprimante du caractère programmable. Remarquer que le canal correspondant qui ne servira plus dans le programme est fermé tout de suite.

. Ligne 150 : Création du format de l'impression

. Ligne 160 : Envoi à l'imprimante du format. Là encore, fermeture du canal dès qu'il ne sert plus.

. Ligne 210 : l'espacement entre lignes et réduit pour imprimer la portée.

. Ligne 250 : cet espacement est augmenté pour les doigtés, et on utilise l'impression formatée. Les CHR $ (29) sont les séparateurs

. Ligne 270 et 280 faire attention que dans "O" l'espace est un espace « shifté »

. Ligne 290 : usage du caractère programmable défini ligne 120.

. Ligne 320 : remise de l'imprimante à un espacement normal pour d'autres calculs.

Ce programme peut s'adapter pour tout instrument à vent.

M. Deconinck

```
100 REM PORTEE MUSICALE & FLUTE A BEC
110 OPEN4,4:OPEN1,4,1:OPEN5,4,6
120 A$=CHR$(15)+CHR$(9)+CHR$(3)+CHR$(15)+CHR$(3)+CHR$(0)
130 OPEN5,4,5:PRINT#5,A$:CLOSE5:REM CARACTERE PROGRAMMABLE
150 C$="  AR  ":C$=C$+C$:C$=C$+C$
160 OPEN2,4,2:PRINT#2,C$:CLOSE2:REM FORMAT IMPRESSION
170 PRINT"■■■■■ PORTEES MUSICALES ■■■■■"
180 INPUT"COMBIEN DE PORTEES";N
190 FOR I=1TON
200 REM OPERATION PORTEE
210 PRINT#5,CHR$(12);:REM ESPACEMENT REDUIT DES LIGNES
220 FOR I=0TO3:PRINT#4:NEXT
230 FOR I=0TO4:FOR J=0TO7:PRINT#4,"_";:NEXT:PRINT#4:NEXT
240 FOR I=0TO4:PRINT#4:NEXT
250 PRINT#5,CHR$(29);:REM ESPACEMENT PLUS FORT A IMPRESSION TABLATURE
260 FOR J=0TO9:PRINT#1,"O";CHR$(29);:NEXT:PRINT#1
270 FOR I=0TO2:FOR J=0TO9:PRINT#1," ";CHR$(29);:NEXT:PRINT#1:NEXT:PRINT#1
280 FOR I=0TO1:FOR J=0TO9:PRINT#1,"/";CHR$(29);:NEXT:PRINT#1:NEXT
290 FOR I=0TO1:FOR J=0TO9:PRINT#1,CHR$(254)CHR$(254)CHR$(29);:NEXT:PRINT#1
300 NEXT:PRINT#1:PRINT#1
310 NEXT
320 PRINT#5,CHR$(24);:REM REMISE IMPRIMANTE INTERLIGNE CLASSIQUE
330 CLOSE1:CLOSE4:CLOSE5
READY.
```

Copyright M. Deconinck et L'Ordinateur Individuel.